ARCHI2 - Compte-rendu du TME1

Nicolas Phan

pour le 17 Janvier 2018

Table des matières

1 Automate du composant PibusSimpleRam

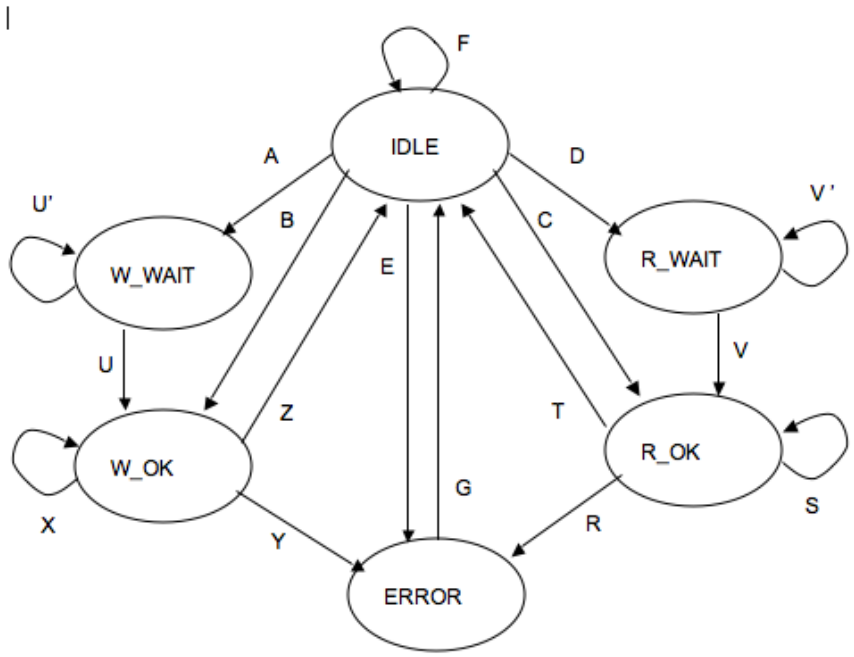

FIGURE 1 – Graphe de la MAE du composant RAM

1.1 Question C1

Nom	Transition
A	SEL.ADR_OK.$\overline{\text{READ}}$.DELAY
B	SEL.ADR_OK.$\overline{\text{READ}}$.$\overline{\text{DELAY}}$
C	SEL.ADR_OK.READ.$\overline{\text{DELAY}}$
D	SEL.ADR_OK.READ.DELAY
E	SEL.$\overline{\text{ADR\_OK}}$
F	$\overline{\text{SEL}}$
G	1
U	$\overline{\text{GO}}$
U'	GO
V	$\overline{\text{GO}}$
V'	GO
X	SEL.ADR_OK.$\overline{\text{READ}}$
Y	SEL.($\overline{\text{ADR\_OK}}$ + READ)
Z	$\overline{\text{SEL}}$
R	SEL.($\overline{\text{ADR\_OK}}$ + $\overline{\text{READ}}$)
S	SEL.ADR_OK.READ
T	$\overline{\text{SEL}}$

TABLE 1 – Fonctions de transition de la MAE de SimpleRam

1.2 Question C2

	ACK_EN	ACK_VALUE	DT_EN	MEM_CMD
IDLE	0	WAIT	0	NOPE
R_WAIT	1	WAIT	0	READ
R_OK	1	READY	0	READ
W_WAIT	1	WAIT	1	WRITE
W_OK	1	READY	1	WRITE
ERROR	1	ERROR	0	NOPE

TABLE 2 – Valeurs des signaux de sortie de la MAE de SimpleRam

2 Automate du composant PibusSimpleMaster

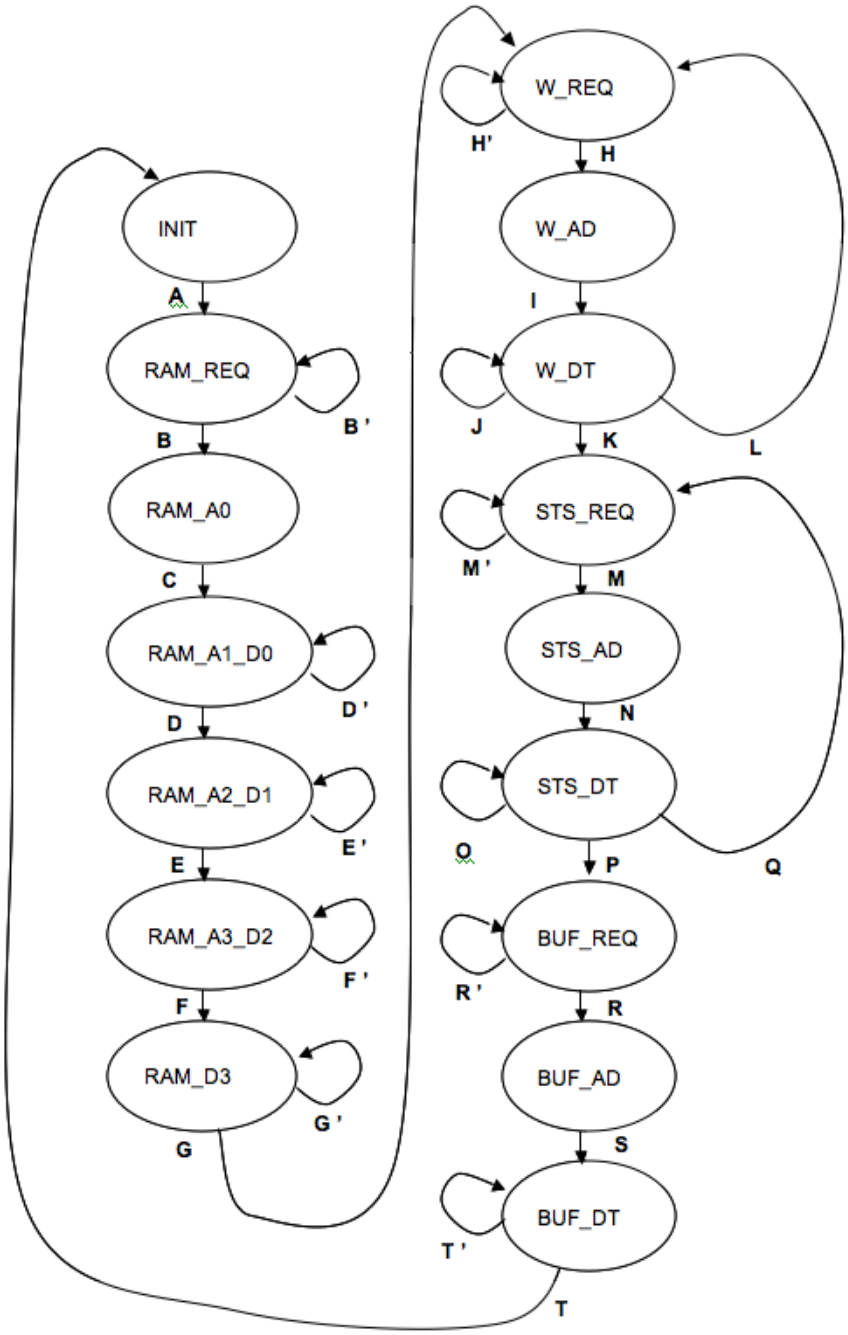

FIGURE 2 – Graphe de la MAE du composant Master

2.1 Question D1

Nom	Transition
A	1
B'	$\overline{\text{GNT}}$
B	GNT
C	1
D'	RDY
D	$\overline{\text{RDY}}$
E	RDY
E'	$\overline{\text{RDY}}$
F	RDY
F'	$\overline{\text{RDY}}$
G	RDY
G'	$\overline{\text{RDY}}$
H'	$\overline{\text{GNT}}$
H	GNT
I	1
J	$\overline{\text{RDY}}$
K	RDY.LAST
L	RDY.$\overline{\text{LAST}}$
M'	$\overline{\text{GNT}}$
M	GNT
N	1
O	$\overline{\text{RDY}}$
P	RDY.$\overline{\text{NUL}}$
Q	RDY.NUL
R'	$\overline{\text{GNT}}$
R	GNT
S	1
T'	$\overline{\text{RDY}}$
T	RDY

TABLE 3 – Fonctions de transitions de la MAE de SimpleMaster

2.2 Question D2

	REQ	CMD_EN	ADR_VALUE	READ_VALUE	LOCK_VAL	DT_EN
INIT	0	0	X	X	X	0
RAM_REQ	1	0	X	X	X	0
RAM_A0	0	1	ram_base	1	1	0
RAM_A1D0	0	1	ram_base + 4	1	1	0
RAM_A2D1	0	1	ram_base + 8	1	1	0
RAM_A3D2	0	1	ram_base + 12	1	1	0
RAM_D3	0	0	X	X	0	0
W_REQ	1	0	X	X	X	0
W_AD	0	1	seg_tty_base	0	0	0
W_DT	0	0	seg_tty_base	0	0	1
STS_REQ	1	0	X	X	X	0
STS_AD	0	1	seg_tty_base + 4	1	0	0
STS_DT	0	0	seg_tty_base + 4	1	0	0
BUF_REQ	1	0	X	X	X	0
BUF_AD	0	1	seg_tty_base + 8	1	0	0
BUF_DT	0	0	X	X	X	0

TABLE 4 – Valeurs de sortie de la MAE de SimpleMaster

3 Automate du composant PibusSegBcu

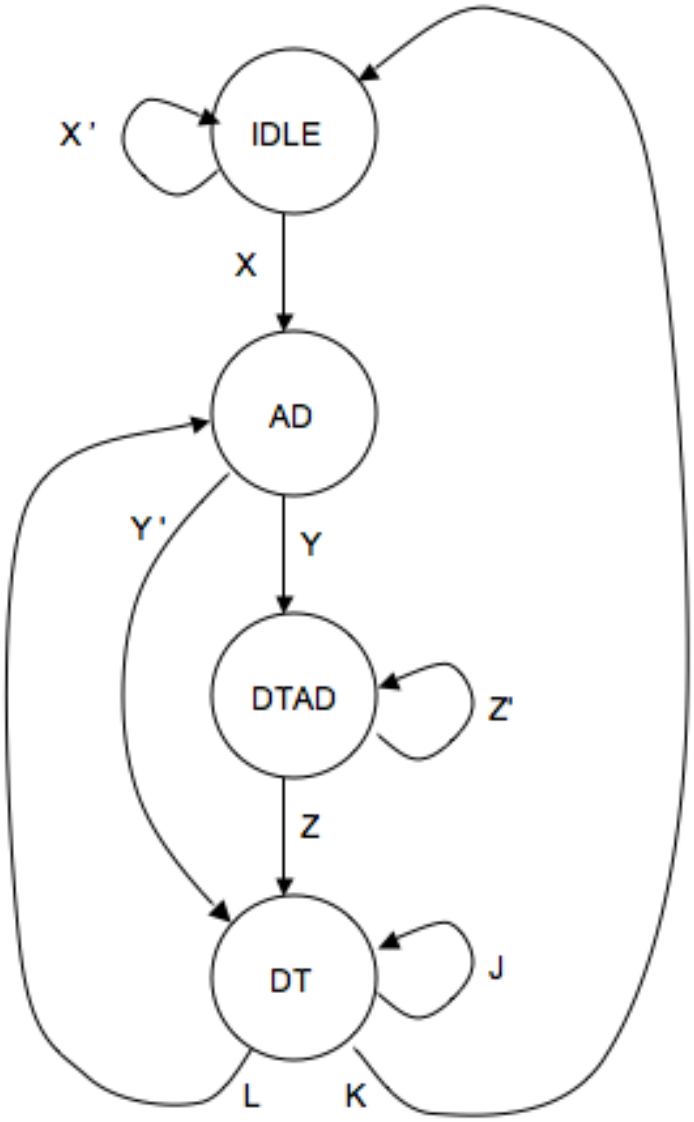

FIGURE 3 – Graphe de la MAE du composant BCU

3.1 Question E1

Nom	Transition
X	$\overline{\texttt{REQ}}$
X	REQ
Y	LOCK
Y	$\overline{\texttt{LOCK}}$
Z	LOCK + $\overline{(\texttt{ACK!} = \texttt{WAIT}}$
Z	$\overline{\texttt{LOCK}}$. (ACK != WAIT)
J	$\overline{(\texttt{ACK!} = \texttt{WAIT})}$
K	(ACK != WAIT) . $\overline{\texttt{REQ}}$
L	(ACK != WAIT) . REQ

TABLE 5 – Fonctions de transition de SegBcu

3.2 Question E2

	GNT	SEL0	SEL1
IDLE	REQ	0	0
AD	1	$A \in zoneRAM$	$A \in zoneTTY$
DTAD	0	$A \in zoneRAM$	$A \in zoneTTY$
DT	REQ	$A \in zoneRAM$	$A \in zoneTTY$

TABLE 6 – Valeurs de sortie de SegBcu

3.3 Question E3

Pour que les bus respectant le protocole PIBUS effectuent les transactions le plus rapidement possible, le protocole spécifie que dans certains cas, une phase d'une transaction peut commencer avant qu'une autre phase de la transaction précédente soit terminée : le bus est pipeliné. Ici, le bus peut entamer une phase d'allocation pendant la phase de réponse de la transaction précédente, c'est pour cela qu l'allocation est réalisée aussi dans DT.

4 Modélisation de l'architecture matérielle

4.1 Question F1

```
1  PibusSegBcu          bcu     ("bcu", segtable, 1 , 2, 100);
2  PibusSimpleMaster    master  ("master", SEG_RAM_BASE, SEG_TTY_BASE);
3  PibusSimpleRam       ram     ("ram"  , 0, segtable, ram_latency, loader);
4  PibusMultiTty        tty     ("tty"  , 1, segtable, 1);
```

4.2 Question F2

```
1  master.p_ck                (signal_ck);
2  master.p_resetn            (signal_resetn);
3  master.p_gnt               (signal_gnt_master);
4  master.p_req               (signal_req_master);
5  master.p_a                 (signal_pi_a);
6  master.p_opc               (signal_pi_opc);
7  master.p_read              (signal_pi_read);
8  master.p_lock              (signal_pi_lock);
9  master.p_d                 (signal_pi_d);
10 master.p_ack               (signal_pi_ack);
11 master.p_tout              (signal_pi_tout);
```

```
12
13    ram.p_ck                    (signal_ck);
14    ram.p_resetn                (signal_resetn);
15    ram.p_sel                   (signal_sel_ram);
16    ram.p_a                     (signal_pi_a);
17    ram.p_read                  (signal_pi_read);
18    ram.p_opc                   (signal_pi_opc);
19    ram.p_ack                   (signal_pi_ack);
20    ram.p_d                     (signal_pi_d);
21    ram.p_tout                  (signal_pi_tout);
```

1. **Modélisation** : Cela consiste en la description d'un modèle du processeur,

La Figure **??** résume le flot de travail et les outils utilisés pour les étapes de Syntèse, Placement et Routage.

$$\sum_{\substack{k\in[[0,4]] \\ \texttt{shift\_value(k)}=1}} 2^k \quad = \texttt{shift\_value}$$